GMPLS control plane extensions in support of flex-grid enabled elastic optical networks

Turus, Ioan; Fagertun, Anna Manolova; Dittmann, Lars

Published in:
Proceedings of OPNETWORK 2013

Publication date:
2013

GMPLS control plane extensions in support of flex-grid enabled elastic optical networks

Ioan Turus, Anna Manolova Fagertun, and Lars Dittmann
Technical University of Denmark
2800 Kgs. Lyngby, Denmark
E-mail: iotu@fotonik.dtu.dk

Abstract
We develop a GMPLS control plane handling flex-grid enabled Elastic Optical Networks. The implementation is realized in OPNET Modeler event driven simulation tool with focus on developing and implementing extensions for the GMPLS protocol suite. RSVP-TE extensions address the reservation of generalized labels format and enable enhancements for the wavelength selection procedures. OSPF-TE enables the creation of spectrum databases based on novel LSA sub-TLV attributes capable of advertising spectrum status. Based on the implemented extensions, we propose and evaluate advanced distributed spectrum allocation schemes and strategies for dynamic routing algorithms in support of flex-grid optical networks.

1. Introduction

Network operators are facing today various challenges in the core optical networks with regards to providing and handling capacity. This is due to the continuously increasing capacity demand, of about 2dB per year according to [1], which faces the current rigid optical network deployments. The main solution that operators use for increasing the provided capacity is to deploy more fibers, extend the number of channels within one fiber or use higher modulation formats for the deployed optical channels. However, none of these represent a long term solution and they are not flexible enough to cope with the scenarios where capacity demand experiences high peaks which are significantly higher than the time average of the demanded capacity.

Elastic Optical Networks (EO-Net) represent a solution which enables elasticity at various levels in the deployment of optical resources. EO-Net considers the variation of different parameters of an optical connection such as symbol-rate, modulation format, Forward Error Correction (FEC) codes and spectrum assignment. Based on the flexibility of the optical connection parameters, the network is able to scale up or down resources according to the demands' properties and it allows operators to deploy a more efficient network. Considering the EO-Net architecture, the network does not need to be planned for peak-traffic conditions but ideally it can follow the average of the traffic demands.

In order to enable the elasticity features, one of the first challenges is to provide flexibility in handling spectrum resources. Currently, Wavelength Switched Optical Networks (WSON) architecture follows the standard ITU-T grid specification [2] which defines the 50 GHz constant channel spacing. In order to enable flexibility in the spectrum assignment, ITU-T defines in the edition 2.0 of [2] an extended granularity in the channel spacing, down to 6.25 GHz. This represents the key point of the so called Flex-grid networks which are associated to the newly defined SSON (Spectrum Switched Optical Networks) architecture [3]. However, by increasing the granularity of the spectrum grid, the complexity of running such network is increased as well and this determines more requirements for the control plane which is running the SSON network.

GMPLS is considered for running SSON based core optical networks as an evolution from WSON networks where GMPLS can already be considered a mature control plane solution. Thus, in order to enable GMPLS to run flex-grid networks, GMPLS protocol suite (OSPF-TE and RSVP-TE) requires a number of extensions to adapt it to the new flexible environment.

This paper presents a detailed OPNET Modeler implementation of the RSVP-TE and OSPF-TE with the required extensions which enable the control of a Flex-grid based Elastic Optical Network. Thanks to the Flex-grid enabled distributed GMPLS control plane, different signaling and routing strategies can be investigated and the control plane efficiency is evaluated in various scenarios. The remainder of this paper is organized as follows: In Section 2 Flex-grid architecture and parameters are described. In Section 3 GMPLS control plane aspects will be analyzed considering the required extensions for enabling flex-grid for both OSPF-TE and RSVP-TE, as well as proposing signaling and routing strategies for improving the control plane efficiency. In Section 4 the OPNET Modeler implementation is presented. Section 5 details the investigated scenarios and results which evaluate the performance of the proposed architecture. The paper ends with concluding remarks regarding the OPNET Modeler implementation and its benefits for further investigating flex-grid and elastic optical networks technology.

2. Flex-grid based Elastic Optical Network (EO-Net)

The Flex-Grid concept considers lower channel spacing compared to the currently used 50 GHz grid. ITU-T proposes different channel spacing values such as 25, 12.5 and 6.25 GHz. The lower bound limit of 6.25 GHz is chosen by taking into account physical limitations and the cost of the current optical filters and Wavelength Selective Switches (WSSs).
The parameters describing the Flex-grid architecture are the following:

- Central Frequency Granularity (CFG) – defines the spacing between two consecutive central frequencies (e.g. 6.25 GHz in Figure 1 for flex-grid example)
- Spectrum Slice Width (SSW) Range – defines the minimum and the maximum size of a Spectrum Slice (e.g. minimum 6.25 GHz, maximum 200 GHz)
- Spectrum Slice Granularity (SSG) – defines the step which gives the possible values for the Spectrum Slice width (e.g. 12.5 GHz step/granularity)

Every Spectrum Slot is identified by its corresponding central frequency. The flexibility in flex-grid environment is provided by the possibility of defining Spectrum Slices which can group a number of contiguous Spectrum Slots. A spectrum slice is defined by using two parameters:

- spectrum slice central frequency (parameter *n*)
- spectrum slice width (parameter *m*)

As shown in Figure 1, the orange slice is described by a central frequency n=-5 and width m=1, while the blue slice is described by n=2 and a width m=4.

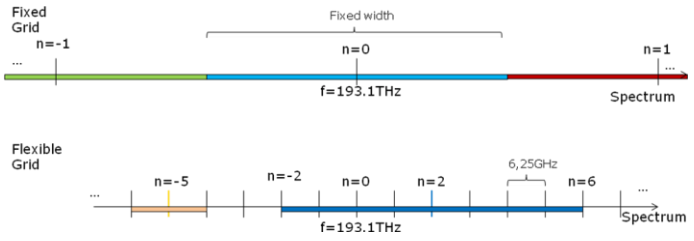

Figure 1: Fixed-grid and Flex-grid architecture

3. GMPLS control plane in support of Flex-grid EO-Net

3.1 RSVP-TE extensions

RSVP-TE is responsible within GMPLS protocol suite for signaling aspects. From a flex-grid perspective, RSVP-TE is required to handle spectrum slots and slices instead of wavelengths. Thus, the label concept has to be generalized to a format which allows the control plane to handle both the fix-grid and the flex-grid channels. As shown in Figure 2, the WSON label format is extended, as suggested in [4], to include a definition for the grid type (e.g. DWDM), the channel spacing or central frequency granularity (e.g. 5 – for 6.25 GHZ) and the index of the central frequency of the slice (*n*).

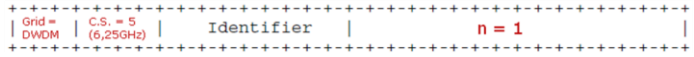

Figure 2: The generalized label format

Moreover, considering the fact that a slice can have variable size (which was not the case for fix-grid channels), apart from generalized labels, RSVP-TE has to carry as well in the *PATH* message the size of the requested slice. The *m* parameter will be included in the *Tspec* object with the format shown in Figure 3.

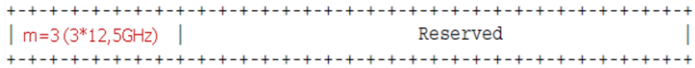

Figure 3: The Spectrum Slice width in *Tspec* object format

RSVP-TE operation will start once the Path Computation engine provides a request for a certain slice size on a predefined path. RSVP-TE will use a *PATH* message to carry the *Tspec* object from source to destination and check in every node all the available Spectrum Slots/Slices that are available and can handle the requested slice size. For every possible set of Spectrum Slots or Slices which are valid for serving the requested slice, the central frequency will be added to the Available Label Set object as shown in Figure 4. The destination node receives all the common central frequencies available to be reserved along the path and based on a local policy decision it will choose one of them and will signalize it back to the source by using the *Resv* message which performs the actual reservation of the specified slice node by node.

The signaling mechanism depicted in Figure 4 describes the distributed GMPLS architecture and the main difference in various implementations is provided by the local policy decision.

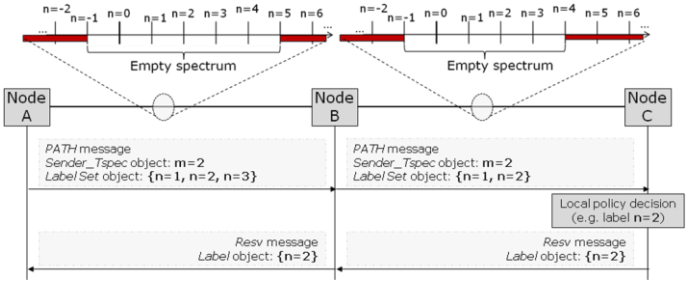

Figure 4: Distributed RSVP-TE signaling in flex-grid networks

For the local policy decision, as part of our work done in [5], we consider three methods:

- *First-Fit*, selecting always the first available central frequency in the label set;
- *Random Assignment*, choosing a random central frequency;
- *Mixed-Fit*, switching between First-Fit and Last-Fit methods consecutively in every node;

3.2 OSPF-TE extensions

OSPF-TE is responsible within GMPLS protocol suite for the routing aspects. In case of flex-grid networks OSPF-TE has to be able to advertise changes that occur in the availability of the spectrum slots. In order to enable this, OSPF-TE Link State Advertisements (LSA) packets will include extended sub-TLVs which can address the status of the spectrum slots.

There are two options we propose in [6] for the format of spectrum slots availability. First option is to extend the Label Set object format for the Inclusive List as shown in Figure 5, and add the state of the slice – free or busy.

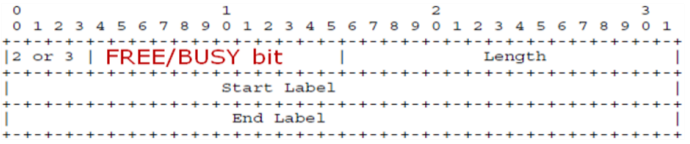

Figure 5: Inclusive List format for Flex-grid Label Set Object

The second option we evaluate is the bitmap encoding instead of Inclusive List for the Label Set format as shown in Figure 6.

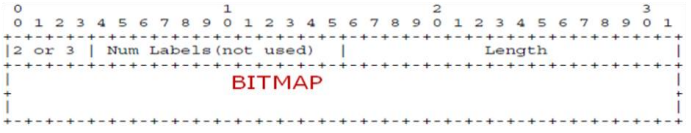

Figure 6: Bitmap encoding format for Flex-grid Label Set Object

An important aspect for the OSPF-TE configuration is the *MinLSInterval* timer value which specifies the minimum time interval where a node cannot send consecutive LSA advertisements. The timer value determines the speed of convergence of the information along the network as well as the load on the control plane.

The information from the LSA advertisements is gathered at every node in a so called Spectrum Database which contains the overall picture of the spectrum status on all the links along the network. This database will serve the Path Computation mechanism in taking optimal decisions for selecting the path.

The Path Computation mechanism will consider a number of routing strategies with the purpose of optimizing the path selection. OPNET Modeler uses by default a method based on Dijkstra graph where the shortest path first algorithm is selected in terms of number of hops. However this is not always the most optimal solution in flex-grid networks. In [6], we propose two strategies called *relaxation graph* and *weighted graph* with the goal of reducing the overall connection blocking.

The *relaxation graph* strategy described by the pseudo-code in Figure 7 is looking into disabling in the source node the links that cannot cope with the requested slice size. Thus, every time a slice is requested, the source node checks the Spectrum Database for all the links in the network if a link has at least one island of *m* contiguous spectrum slots. If no such island can be found it means that that link cannot handle the requested slice and it is temporarily disabled in the Dijkstra graph of the source node. After all the links are verified, the source node performs the Shortest Path First algorithm on the updated Dijkstra graph.

```
W = total number of Spectrum Slots (or central frequencies) in a fiber
L = total number of links/fibers in the network
m = requested slice size (expressed as number of Spectrum Slots)

for i = 0 to (L-1)
        {
        for j = 0 to (W-1-m)
                if (slots from j to j+m are free)
                        break;
        Disable in Dijkstra graph link i;
        }
```

Figure 7: Relaxation graph mechanism pseudo-code

The *weighted graph* strategy described in the pseudo-code in Figure 8 considers to what extent the links are occupied.

```
W = total number of Spectrum Slots (or central frequencies) in a fiber
L = total number of links/fibers in the network
m = requested slice size (expressed as number of Spectrum Slots)

for i = 0 to (L-1)
        {
        count = 0;
        for j = 0 to (W-1)
                if (slot_j is occupied)
                        count++;
        Assign in Dijkstra graph weight count for link i;
        }
Perform shortest path first (SPF) algorithm on weighted
Dijkstra graph
```

Figure 8: Weighted graph mechanism pseudo-code

Thus, the algorithm checks the links occupancy in every source node every time a new slice/connection is requested. The verification is done by using the information from the Spectrum Database. For every link, the number of occupied spectrum slots is computed and the resulted value is assigned to the Dijkstra graph as weight for the corresponding edge (or link). After all edges are updated with the up-to-date weights, the Shortest Path Algorithm is executed and it selects the path with the lowest weight, prioritizing in this way the least loaded links.

4. OPNET Models of the GMPLS control plane for the Flex-grid based EO-Net

The network simulation model has been developed over three layers: network model, node model and process model.

4.1 Network and node models

The OPNET Modeler implementation took into account two different network models as presented in Figure 9 and Figure 10 for the sake of diversity of the network characteristics.

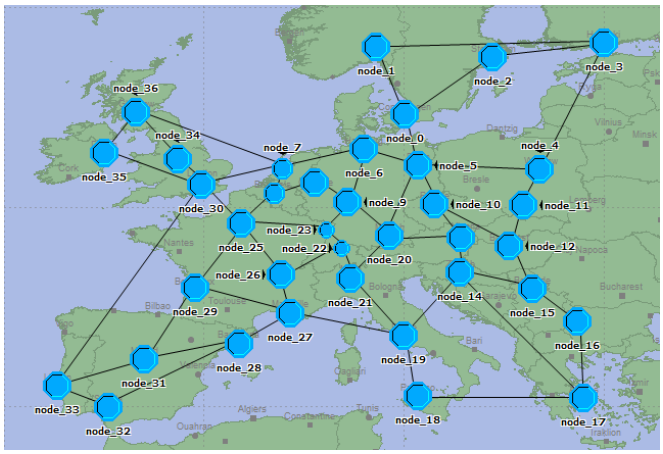

Figure 9: Network model based on COST 266 topology

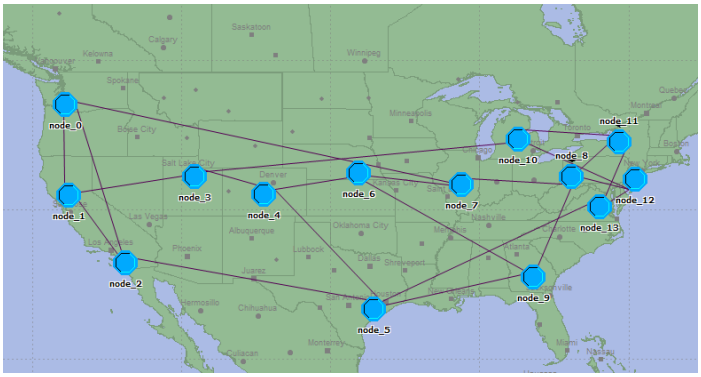

Figure 10: Network model based on NSF topology

In the current implementation and simulation scenarios the COST 266 topology was used for evaluating the RSVP-TE enhancements while the NSF topology was used to evaluate the OSPF-TE extensions.

The node model is shown in Figure 11 and it consists of a connection generator (*ReqGen*), a routing module (*Routing_module*), an instance of OSPF-TE, an instance of RSVP-TE and the associated seven receivers (*pr_x*) and transmitters (*pt_x*).

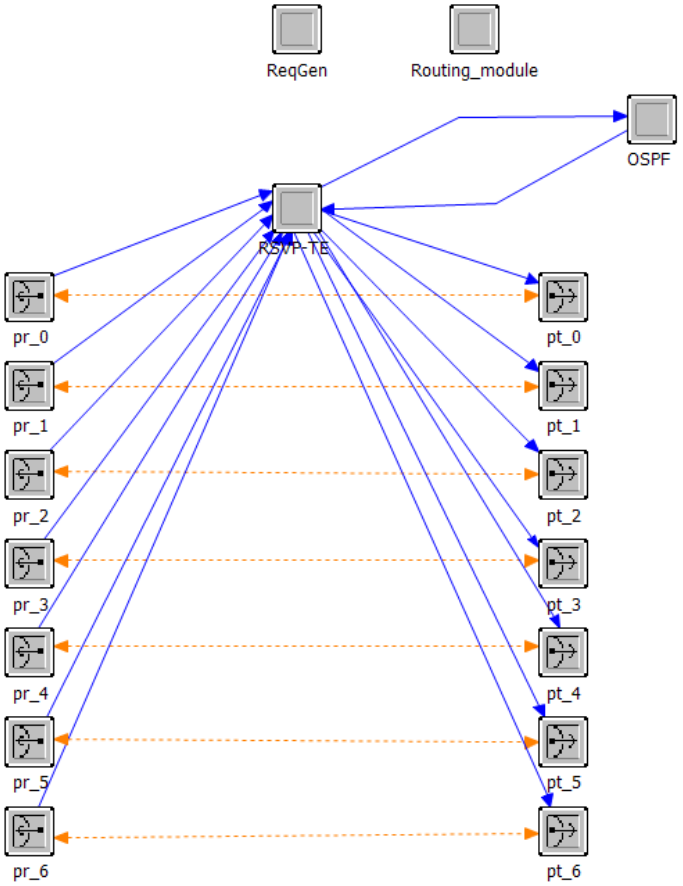

Figure 11: Node model

The connection generator is responsible to create elastic connections with specific properties corresponding to the elastic transponder capabilities described in [7]. The OSPF model is responsible to handle LSA packets and update the Spectrum Database as well as handling the advanced routing strategies. The routing model handles the routing table by associating to every directly connected node the corresponding interface. The RSVP-TE model is responsible for providing a path and handling the *PATH*, *Resv* and the rest of signaling messages required to reserve resources along the path.

Connection generator process model

The Finite State Machine (FSM) describing the *ReqGen* model is shown in Figure 12. The *Init* state handles the variable initialization, statistics definition, attributes retrieval and ends with changing state to *Idle*. In *Idle* state self-interrupts are scheduled according to an exponential distribution defining the Mean Interval Time (MIT) between consecutively generated connections.

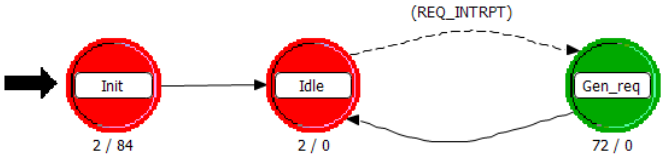

Figure 12: Connection generator model

The parameters describing the connection generator are shown in Table 1. The bandwidth for every connection is uniformly distributed between 0 and 100 Gbps and a round-up is made to the closest higher level of bit-rate which is available in an elastic transponder according to [7]. 1 bit per symbol is considered and the 4 available bit-rate levels are 25, 50, 75 and 100 Gbps which results in 25% probability of obtaining a connection in one of the four bit-rate or bandwidth levels.

Variable	Value
Flow destination	Uniform distribution (total number of nodes)
Bandwidth	Uniform distribution (0…100Gbps)
Duration	Exponential distribution (mean of given <dur> variable)
Mean Interval Time	Exponential distribution (mean of given <mit> variable)

Table 1: Connection generator parameters

The *Gen_req* state is also responsible to create remote interrupts which send the connections parameters to the RSVP-TE module.

4.1 Process models

The process models used in the current implementation represent an extension of the OSPF, RSVP-TE and routing models from [8]. The models were adapted to efficiently handle flex-grid particularities and their functionality, including the flex-grid specific changes, is described below.

Routing module

The FSM for the routing module is depicted in Figure 13. The model starts with the *Init* state where attributes for the routing or signaling strategies, as well as for the flex-grid architecture are retrieved. The *Init* state is also responsible to associate a Dijkstra graph (made by vertices and corresponding edges) to the implemented topology by creating a routing table. Based on the Dijkstra graph created at this moment, the path computation will be performed for every incoming connection. Once the routing table is constructed, the routing module also triggers the OSPF and RSVP-TE modules by sending remote interrupts.

If the node is enabled in the network, the process moves to the *Operational* state, otherwise it ends in the *End_Sim* state. The *Disable_Edge* state is used for situations where all the resources on one link (spectrum slots) are used and the link/edge has to be disabled from the Dijkstra graph or in case where resources become available and the link can be re-enabled. In such situations, the RSVP-TE is performing a remote interrupt for the routing module and identifies the edge using global variables.

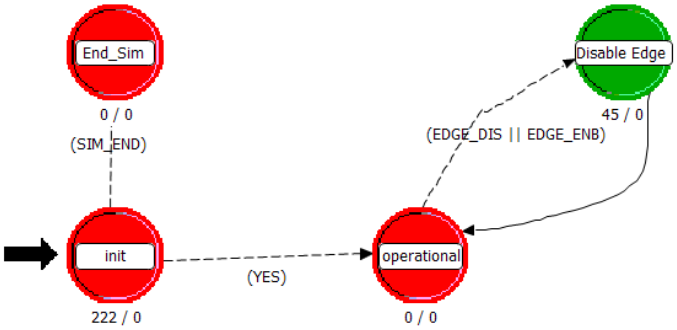

Figure 13: Routing process model

OSPF-TE module

The OSPF-TE module is run once the routing table is initialized in the Routing module and a remote interrupt is received. The role of this module is to be able to create and receive Spectrum LSAs (Extended TE LSAs), update Spectrum Database and confirm the advertisements for the other nodes.

The model implements OSPF version 2 at a simplified format with the purpose of emphasizing the effect of the flex-grid environment in an OSPF/GMPLS based control plane. According to the flex-grid architecture, whenever a connection is set-up or released, a number of spectrum slots are occupied or released as well. This information is required for the rest of the nodes in the network so they can update their Spectrum Databases and provide optimal path computation. Thus, whenever a slice or set of spectrum slots is changing the status, the OSPF module has to create a Spectrum LSA and flood it to its neighbors (according to the OSPF specifications). Similarly, whenever a node receives a Spectrum LSA, it will update its own Spectrum Database and will confirm the receipt to the corresponding node.

The implementation is extended from the one in [8] with a number of additions. According to the OSPF-TE FSM shown in Figure 14, the *Init* state is responsible for variable initialization and attributes retrieval. In state *Begin* the initialization of the neighbors list and links is performed and the process moves to the *Exchange* core state. From this state, the process can switch to one of the three *Create* processes whenever the OSPF process receives an interrupt because of a change in the spectrum of the links directly connected to this node. Depending on the model of Spectrum LSA format that is used (as described in Section 3.2) the process goes to *Create2* or *Create3* states. The only difference between the two states is the format of the sub-TLV that carries the spectrum state information. In the *Create (2 or 3)* state, the Spectrum LSA is created and it is flooded to all neighbors. A sensitive parameter implemented here is the *MinLSInterval* timer which according to [9] defines the minimum period between creating two consecutive LSAs. In case a request arrives for a new LSA before *MinLSInterval* timer passes, the LSA creation is ignored. *Create1* state handles the creation of another type of LSA (Flex-Grid properties LSA) at the beginning of the simulation. This type of LSA is responsible for advertising the flex-grid capabilities and parameters specific for every node. However, in this implementation, this type of LSA is not used because we consider that all nodes in the network have the same flex-grid parameters.

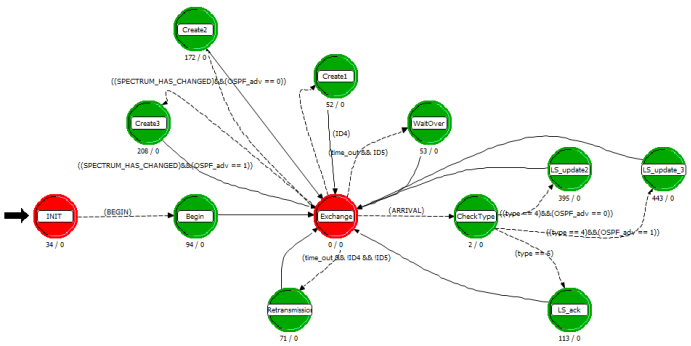

Figure 14: OSPF-TE process model

The *WaitOver* and *Retransmission* states are similar to the implementation in [8] while the *LS_Update2* and *LS_Update3* are performing the update of the Spectrum Database according to the information received from the Spectrum LSA. The two types of *LS_Update* refer to the two types of Spectrum LSA format. Once the LSAs are received and the updates are performed, the acknowledgement messages (ACKs) are sent back to the originating nodes.

When the *CheckType* state identifies that a received packet is not a Spectrum LSA but an ACK packet, it sends it to the *LS_ACK* state which is responsible for updating the list of received ACKs. This is done for verifying if ACK messages are received from all neighbors and if this is not the case, retransmission should be taken into account.

RSVP-TE module

The RSVP-TE module is responsible for handling every incoming connection. The module deals with the specific *PATH* and *Resv* messages as well as with the rest of RSVP messages describing different errors in reserving resources. For each connection, an associated child process is created and this will track the progress of the connection from signaling until tearing-off when the resources and the child process are removed.

RSVP-TE module is described by the FSM implementation in Figure 15. The *Init* state performs a mapping between the simulated model links and nodes and the Dijkstra graph (edges and vertices) associated to this node with the purpose of simplifying the process of handling and updating edges and weights. Once the module receives the remote interrupt from the Routing module, the process moves to the *Idle* state. From here, the process uses the *Req_Handle* state for every incoming connection request as well as receiving a *PATH*, *Resv* or an error message. Also, in the *Req_Handle* state, the actual Path computation process is performed which can take into account either the Shortest Path algorithm or the *Relaxation graph* and *Weighted* graph procedures for advanced routing strategies. The *Weighted graph* strategy is performed by always updating the weights for every edge in the Dijkstra graph before performing path computation while the *Relaxation* Graph strategy is done by temporarily enabling or disabling the edges according to the requested slice size.

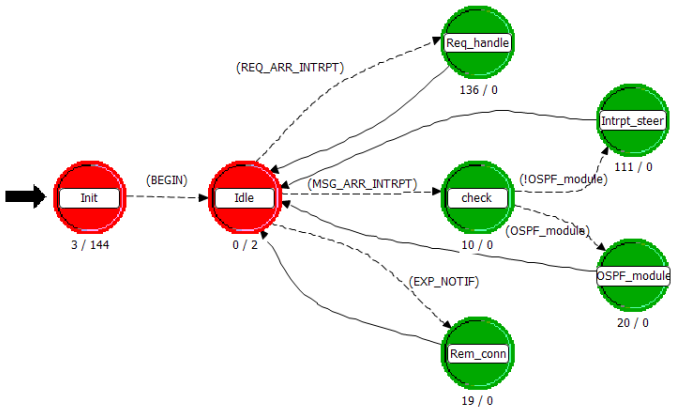

Figure 15: RSVP-TE process model

Whenever a new connection is requested and a new child process is created, the number of required Spectrum Slots (slice size) is computed by taking into account the flex-grid attributes that are associated to this node (e.g. Central Frequency Granularity).

The process also handles the receipt of packets or messages and this is handled by checking the message type in the *check* state. In case it is an OSPF message (LSA or ACK), this is forwarded on the interface towards the OSPF module, otherwise the message is passed to the child process which will consider it depending on the current state of the connection.

The *Rem_conn* state is responsible to release the resource in the case a connection expires. The OSPF child process also contains the implementation of the Spectrum Assignment strategies as well as the particular implementation of the RSVP mechanism with regards to the flex-grid architecture, such as: collecting spectrum slots, verifying that spectrum slots are contiguous, considering only the central frequency and the size of a slice to identify a connection etc.

4.2 Attributes and statistics

In order to make the model and the scenario configuration more flexible, a number of attributes were defined so they can be adjusted according to the user's demand and they are described in Table 2.

Attribute	Possible value
CFG (Central Frequency Granularity)	6.25; 12.5; 25; 50 [GHz]
SSG (Spectrum Slot Granularity)	12.5; 25; 50 [GHz]
OSPF advertising procedure	Inclusive Label Set Bitmap encoding
OSPF routing algorithm	Shortest Path First Shortest Path First on weighted Dijkstra graph
Relaxation graph (strategy)	ON / OFF
Wavelength Assignment	First-Fit, Mixed-Fit, Random Assignment
OSPF Retransmission timer	[integer value]
OSPF MinLSInterval	0, 30, 60, 180 [sec]

Table 2: Network model attributed

Regarding the collected statistics, in this implementation we consider the *blocked connections* which represent the total number of blocked connections in the network and the *network load* which stands for the total number of created LSA advertisement packets in the network.

5. Simulation Results and Analysis

5.1 Signaling scenario

In order to test and evaluate the proposed and implemented extensions for RSVP-TE, we deployed a COST 266 network model as shown in Figure 9.

We evaluate the three different spectrum assignment methods: First-Fit, Random-Assignment and Mixed-Fit. The results in terms of percentage of blocked connections are depicted in Figure 16. It can be observed that First-Fit outperforms the other methods because it packs the spectrum better while minimizing the effects of fragmentation.

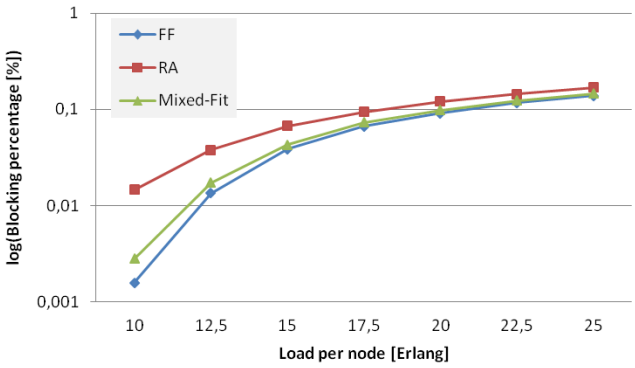

Fig 16: Blocking for different Spectrum Assignment methods

Fragmentation represents an issue for the flex-grid environment because when slices are assigned, there is a possibility that between two assigned neighbor slices there is a gap of free spectrum between them. This scenario can be repeated in a single fiber or link and considering also that those gaps can have small sizes, they cannot handle specific slice requests.

5.2 Routing scenario

For the routing evaluation an NSF network model is deployed as shown in Figure 10. The goal of the implementation is to prove the functionality of the OSPF-TE extensions as well as to emphasize the increased efficiency of the flex-grid architecture based on advanced routing strategies. The results presented further were also submitted and accepted in [8].

The simulation scenario considered different *MinLSInterval* timer value for the LSA updates and verifies the two formats for the LSA Label Set sub-TLV. As shown in Figure 17, if the Inclusive Label Set sub-TLV format is used in the LSA packets, the percentage of the blocked connections is highly dependent on the timer value.

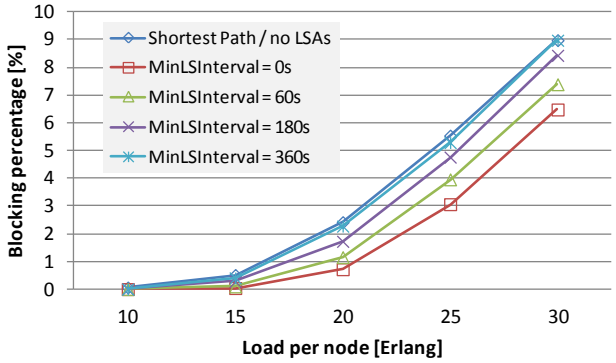

Fig 17: Blocking for different advertising timers in case of using Inclusize sub-TLV format

It can be also observed in Figure 17 that the lowest blocking is achieved for *MinLSInterval* 0 which means that whenever a change in the spectrum is observed on any link connected to that node, an LSA packet is created to advertise that change without waiting for any timer. Moreover, if higher timer values are used (e.g. 360 sec or more), the blocking percentage becomes worse than for the Shortest Path case – which represents the case where no OSPF-TE LSA based information is used for optimizing path computation.

On the other side, in Figure 18 we can observe that when we implement the LSAs using bitmap encoding sub-TLV format, the blocking is less dependent on the *MinLSInterval* timer value. This is because the bitmap encoding sends the whole spectrum information in one fiber and not only the information about specific spectrum slots which get changed as in the previous case. Because of this format, even if the timer value is high, an LSA update covers changes that occurred during the timer period by sending the latest update of the spectrum status in the whole fiber. We can notice in Figure 18 that even for high values of the *MinLSInterval*, the blocking remains at similar low level.

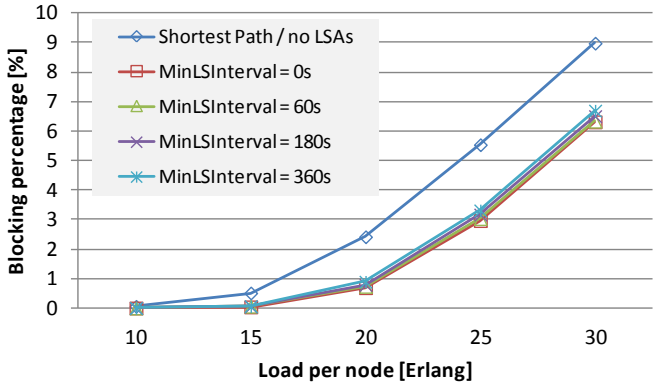

Fig 18: Blocking for different advertising timers in case of using bitmap encoding sub-TLV format

The *MinLSInterval* timer configuration is important because it affects the load that the LSA packets determine on the control plane. At the same time the timer selection is a trade-off between getting up-to-date OSPF-TE LSAs and keeping the control plane load at a reasonable level. As it is shown in Figure 19, if the *MinLSInterval* timer is lower than 180 seconds in this scenario, the control plane load gets very high and unstable. The conclusion is that the bitmap encoding is clearly preferable for advertising flex-grid spectrum slot information in LSA packets.

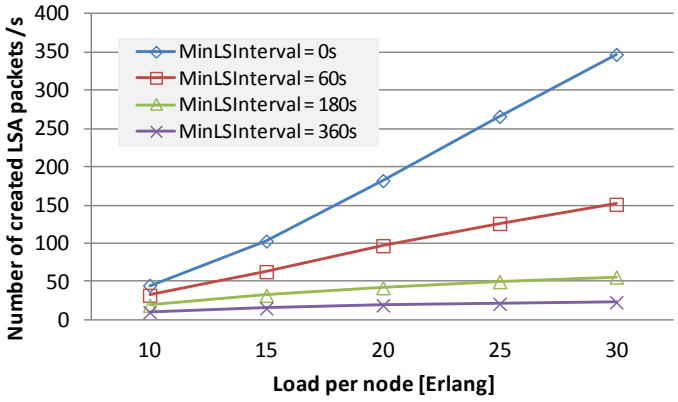

Fig 19: Control plane load due to OSPF-TE overhead

Based on the information gathered from the OSPF-TE LSA advertisements, Spectrum Databases are created containing information about the status of the spectrum on all the links in the network. By using this information in the path computation engine, the source node is able to select an optimal path for the connection, considering the amount of available resources in every link in the network. Figure 20 shows the benefit of using two routing strategies based on the information gathered in the Spectrum Database. In the case of *Relaxation graph* strategy, by only disabling the links that cannot cope with the requested slice, 15% improved blocking can be achieved. In the other case of using *Weighted graph* strategy, up to 70% lower blocking can be achieved for low loaded network (approx. 10 to 30 Erlangs).

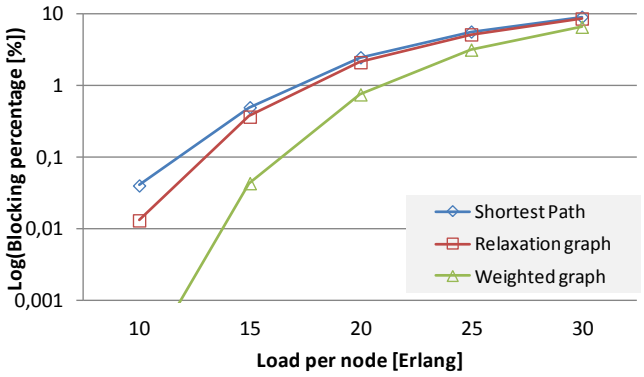

Fig 20: Blocking for advanced routing strategies

Conclusion

The paper presents a novel OPNET Modeler implementation of GMPLS based control plane capable of handling flex-grid elastic optical networks. The model has a flexible design which allows extensions for the OSPF-TE and RSVP-TE not only to support flex-grid but also to enable other flexible parameters specific for elastic optical networks. Throughout this implementation we prove the efficiency of the flex-grid extensions and we enable evaluation for proposed strategies for both routing and signaling aspects.

References

[1] R. W. Tkach "Scaling optical communications for the next decade and beyond" Bell Labs Technical Journalk 14(4), 3-10 2010

[2] ITU-T G.694.1 Recommendation "Spectral grids for WDM applications: DWDM frequency grid"

[3] F. Zhang et al., "Framework for GMPLS and PCE Control of Spectrum Switched Optical Networks", IETF draft 2012

[4] D. King et al., "Generalized labels for the Flexi-Grid in Lambda-Switch-Capable (LSC) Label Switching Routers", IETF draft, 2013

[5] I. Turus et al., "Evaluation of Distributed Spectrum Allocation Algorithms for GMPLS Elastic Optical Networks", submitted to Globecom 2013

[6] I. Turus et al., "Evaluation of Strategies for Dynamic Routing Algorithms in Support of Flex-Grid based GMPLS Elastic Optical Networks", ECOC 2013, We.3.E.5

[7] A. Morea et al., "Efficiency gain from elastic optical networks", Communications and Photonics Conference and Exhibition, 2011

[8] J. Wang et al., ”Energy-efficient routing in GMPLS network”, OPNETWORK 2011

[9] D. Katz, “Traffic Engineering (TE) Extensions to OSPF Version 2”, RFC3630, Sept. 200